## Claim Amendments

Please amend the claims to be as follows.

1. (previously presented) A method of compiling a program to be executed on a target microprocessor with multiple execution units of a same type, the method comprising:

selecting, by a program compiler, one of the execution units for testing;

scheduling, by the program compiler, execution of diagnostic code on the selected execution unit; and

scheduling, by the program compiler, execution of program code on remaining execution units of the same type,

wherein said execution of diagnostic code on the selected execution unit and said execution of program code on the remaining execution units are scheduled to be performed in parallel.

2. (original) The method of claim 1, wherein the selection of the execution unit for testing utilizes an algorithm that assures testing of each of the multiple execution units.

3. (currently amended) The method of ~~claim 2, wherein the algorithm comprises a round-robin type algorithm.~~ claim 1, wherein source code is input into the program compiler, object code is output from the program compiler, and wherein said scheduling is performed by the program compiler prior to execution of the object code by the target microprocessor.

4. (original) The method of claim 1, further comprising:

setting a level of aggressiveness for scheduling the testing of the execution units.

5. (original) The method of claim 4, further comprising:

applying an aggressiveness-dependent algorithm to determine when to schedule all available units for execution of the program code and when to schedule parallel execution of the program code and the diagnostic code.

6. (original) The method of claim 5, wherein a lowest level of aggressiveness comprises turning off said testing.

7. (original) The method of claim 1, wherein the multiple execution units of the same type comprise arithmetic logic units.

8. (original) The method of claim 1, wherein the multiple execution units of the same type comprise floating point units.

9. (original) The method of claim 1, wherein the multiple execution units comprise at least four execution units of the same type integrated onto the microprocessor integrated circuit.

10. (original) The method of claim 1, wherein the scheduled diagnostic code performs diagnostic operations from a test pattern comprising operations with known expected results.

11. (original) The method of claim 10, wherein the scheduled diagnostic code compares an actual result with a known expected result.

12. (original) The method of claim 11, wherein the scheduled diagnostic code jumps to a fault handler if the compared results are different.

13. (original) The method of claim 12, wherein the fault handler includes code to remove a faulty execution unit from use in executing code.

14. (original) The method of claim 12, wherein the fault handler includes code to perform a system halt to prevent data corruption.

15. (previously presented) A computer-readable medium having a program product for execution on a target microprocessor having multiple execution units of a same type integrated thereon, the program product comprising:

microprocessor-executable diagnostic code stored on the computer-readable medium and configured by a program compiler to be executed on a selected execution unit of the multiple execution units; and

microprocessor-executable program code stored on the computer-readable medium and configured by the program compiler to be executed on remaining execution units of the same type,

wherein said diagnostic code and said program code are scheduled to be performed in parallel on the selected execution unit and the remaining execution units, respectively.

16. (previously presented) The computer-readable medium of claim 15, wherein the selected execution unit rotates between the multiple execution units such that each execution unit is tested.

17. (previously presented) The computer-readable medium of claim 15, wherein the multiple execution units of the same type comprise arithmetic logic units.

18. (previously presented) The computer-readable medium of claim 15, wherein the multiple execution units of the same type comprise floating point units.

19. (previously presented) The computer-readable medium of claim 15, wherein the multiple execution units comprise at least four execution units of the same type integrated onto the microprocessor integrated circuit.

20. (previously presented) The computer-readable medium of claim 15, wherein the scheduled diagnostic code performs diagnostic operations from a test pattern comprising operations with known expected results.

21. (previously presented) The computer-readable medium of claim 20, wherein the diagnostic code compares an actual result with a known expected result.

22. (previously presented) The computer-readable medium of claim 21, wherein the diagnostic code jumps to a fault handler if the compared results are different.

23. (previously presented) The computer-readable medium of claim 22, wherein the fault handler includes code to remove a faulty execution unit from use in executing code.

24. (previously presented) The computer-readable medium of claim 22, wherein the fault handler includes code to perform a system halt to prevent data corruption.

25. (previously presented) A computer-readable medium having a program product for execution on a target microprocessor having multiple execution units of a same type integrated thereon, the program product comprising:

microprocessor-executable diagnostic code stored on the computer-readable medium and scheduled by a program compiler to be executed on a selected execution unit of the multiple execution units; and

microprocessor-executable program code stored on the computer-readable medium and scheduled by the program compiler to be executed on remaining execution units at a same time as the diagnostic code is to be executed on the selected execution unit,

wherein the selected execution unit rotates between the multiple execution units such that each execution unit is tested, and

wherein said diagnostic code is further configured to be run in a background type process on a multi-threaded operating system.